FACULTY OF TECHNOLOGY AND ENGINEERING

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

A.Y. 2023-24 [EVEN]

LAB MANUAL

CE266: SOFTWARE ENGINEERING

ID: 22DCE001 , 22DCE006 , 22DCE011 ,
22DCE016 , 22DCE017 , 22DCE018 ,
22DCE020

Practical 2

Aim: Study and compare different software process models and compare them based on cost, simplicity, risk, involvement of user, flexibility, maintenance, integrity, security, re-usability, and requirement.

[Students need to study all models and present GroupWise; particular batch must cover each process models and finally students have to select particular process model for their SGP project with proper assessment and justification.]

Theory:

SDLC Waterfall Model

What is Waterfall Model?

The classical waterfall model which is also known as the linear-sequential life cycle model is an essential software development model which can be understandable from the structure itself. The model is straightforward yet idealistic. When this model was first introduced, it used to be very popular, but time, the new model has come up with a change in features and requirements and hence it is used decidedly less but still a popular one which everyone must know. All the old software has been developed based on this model's life cycle. It is a sequential model which segregates software development into different phases. Each phase is designed with some unique functionality and use. The model was pioneered in the year 1970 by Winston Royce.

Stages of Waterfall Model:

The various phases of the Waterfall Model which are shown below:

1. Requirement Gathering Stage/Feasibility Study
2. Design Stage
3. Built Stage
4. Integration and Test Stage
5. Deployment Stage
6. Maintenance Stage

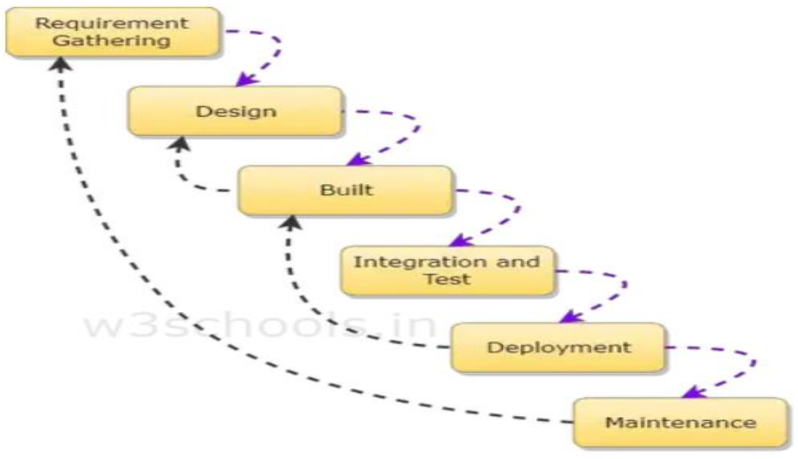


Fig: SDLC Waterfall Model

Requirement Gathering

This phase has the purpose to establish whether it would be monetarily as well as technically practicable to expand the development of software. This has the achievability study with the understanding of the problem as well as determines the diverse potential strategies used for solving the problem.

Design Stage

There is a thorough study of the entire requirement specifications from the first phase, and then the system design is equipped. This phase helps developers to specify hardware as well as the system's requirement which ultimately helps in characterizing the system design as a whole.

Built Stage

This phase is also known as the coding phase of software development where the idea is converted into source code and UI plus UX design using programming language and tools. Hence, every designed module needs to be coded.

Integration and Test Stage

Once the coding of application is done, it is then integrated with all other modules with different functionality. During each step of integration, earlier planned modules are incorporated into the parts included the structure of the software and then the entire system is tested.

1. α Testing: In this testing, the software is tested by the development team, i.e., the developers.
2. β Testing: In this testing, the software is tested by friendly customers and other target users who will use the beta version of your product.
3. Acceptance Testing: Once the application has been distributed, the customer carries out the acceptance test for determining if the product should be accepted as delivered or rejects it for further modification.

Deployment Stage

As all the functional, as well as non-functional tests, are completed, the software is installed in the customer's end or the environment or gets released in the market.

Maintenance Stage

Another important phase of this model is the maintenance model. Updating the product, patching any bugs and errors and developing other essential components as per feedback to make this full software is done in this stage. It is of three types:

1. **Corrective Maintenance:** Corrective maintenance is where the maintenance is done to fix the errors.
2. **Perfective Maintenance:** Perfective maintenance is done where the maintenance is done to increase the efficiency of any system according to customer's requirement.

3. **Adaptive Maintenance:** Adaptive Maintenance is typically necessary for porting your application to a new work environment or porting from one type of OS to another.

Advantages of Waterfall Model:

Clear Structure and Phases: The Waterfall model follows a structured approach with distinct phases such as requirements gathering, design, implementation, testing, deployment, and maintenance. This makes it easy to understand and manage.

Well-Defined Requirements: Since all requirements are gathered upfront in the requirements phase, it provides a clear understanding of what needs to be developed.

Documentation: The model emphasizes documentation at every phase, ensuring that there's comprehensive documentation available for each stage of development, which can be beneficial for maintenance and future enhancements.

Easy to Manage: The linear and sequential nature of the Waterfall model makes it easier to manage as each phase has specific deliverables and a review process.

Disadvantages of Waterfall Model:

Inflexibility: Once a phase is completed, it's challenging to go back and make changes without affecting subsequent phases. This lack of flexibility can be problematic if changes are required later in the project.

Late Testing: Testing is typically done towards the end of the development cycle. This means that defects or issues might only be identified late in the process, leading to higher costs and risks.

Longer Time to Market: Due to its sequential nature, the Waterfall model may result in longer development cycles, delaying the time to market.

Customer Feedback: Since the customer or end-user feedback is generally obtained late in the process, there's a risk of developing a product that doesn't fully meet user needs or expectations.

Applications of Waterfall Model:

Large-Scale Projects with Clear Requirements: The Waterfall model can be suitable for large-scale projects where requirements are well-understood and unlikely to change significantly.

Mission-Critical Systems: For projects where reliability and predictability are crucial, such as developing software for critical infrastructure, the Waterfall model can be beneficial.

Regulated Industries: In industries like healthcare, aerospace, or finance, where regulatory compliance is essential, a structured approach like Waterfall can help ensure that all requirements are met and documented.

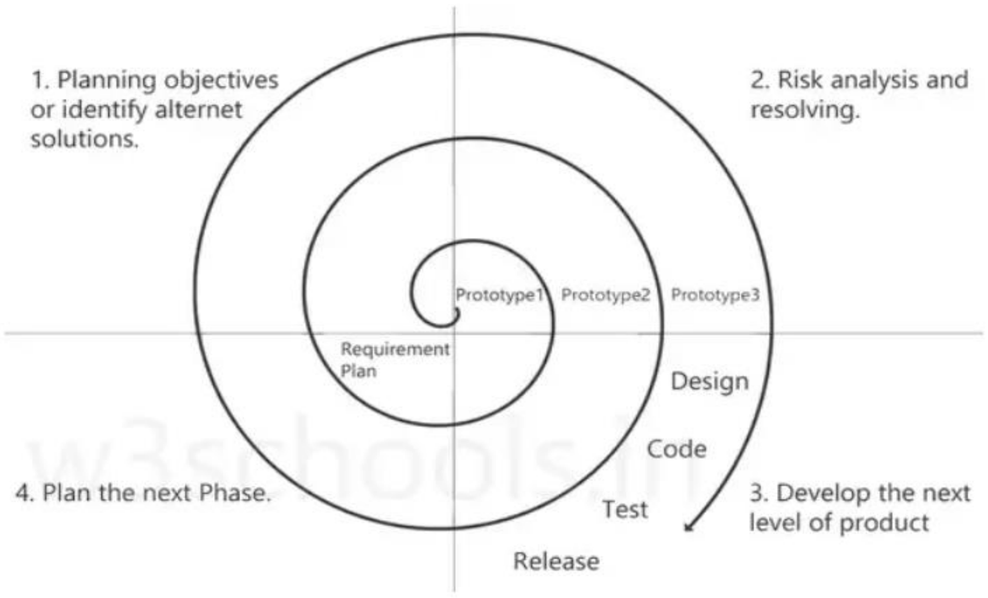
Projects with Stable Technologies: When working with well-established technologies and methodologies where there's a clear understanding of the system requirements, the Waterfall model can be effective.

SDLC Spiral Model

What is Spiral Model?

The popular spiral model is a blend of both iterative development method as well as sequential improvement model, i.e., the waterfall model that is having exceptionally high importance on risk analysis. In this model, the exact number of phases for developing a product varied based on some constraints and by project manager which calculates the project risks. Here the project manager dynamically decides the number of phases and hence play a significant role in the development of a product using the spiral model. The radius in spiral usually shows the expenses or cost needed for project development. The angular dimension shows the development done to date during the recent phase.

Graphical Representation of Spiral Model


1. Planning objectives or identify alternet solutions.
2. Risk analysis and resolving.
Prototype1
Prototype2
Prototype3
Requirement Plan
Design
Code
Test
Release
4. Plan the next Phase.
3. Develop the next level of product
w3schools.in

Different Phases of Spiral Model

The phase of the spiral model has four quadrants, and each of them represents some specific stage of software development. The functions of these four quadrants are listed below:

1. **Planning objectives or identify alternative solutions:** In this stage, requirements are collected from customers and then the aims are recognized, elaborated as well as analyzed at the beginning of developing the project. If the iterative round is more than one, then an alternative solution is proposed in the same quadrant.
2. **Risk analysis and resolving:** As the process goes to the second quadrant, all likely solutions are sketched, and then the best solution among them gets select. Then the different types of risks linked with the chosen solution are recognized and resolved through the best possible approach. As the spiral goes to the end of this quadrant, a project prototype is put up for the most excellent and likely solution.
3. **Develop the next level of product:** As the development progress goes to the third quadrant, the well-known and mostly required features are developed as well as verified with the testing methodologies. As this stage proceeds to the end of this third quadrant, new software or the next version of existing software is ready to deliver.
4. **Plan the next Phase:** As the development process proceeds in the fourth quadrant, the customers appraise the developed version of the project and reports if any further changes are required. At last, planning for the subsequent phase is initiated.

Advantages of Spiral Model:

- **Suitable for large projects:** Spiral models are recommended when the project is large, bulky or complex to develop.
- **Risk Handling:** There are a lot of projects that have un-estimated risks involved with them. For such projects, the spiral model is the best SDLC model to pursue because it can analyze risk as well as handling risks at each phase of development.
- **Customer Satisfaction:** Customers can witness the development of product at every stage and thus, they can let themselves habituated with the system and throw feedbacks accordingly before the final product is made.
- **Requirements flexibility:** All the specific requirements needed at later stages can be included precisely if the development is done using this model.

Disadvantages of Spiral Model:

Complexity: The Spiral Model is more complex compared to other SDLC models like Waterfall. It requires a clear understanding of risk assessment and management, which might be challenging for inexperienced teams.

Costly: Due to its iterative nature and emphasis on risk management, the Spiral Model can be costlier and time-consuming, especially if the project involves multiple iterations.

Not Suitable for Small Projects: For small projects or projects with limited resources, the overhead associated with risk assessment and prototyping might be excessive.

Requires Expertise: Effective implementation of the Spiral Model requires expertise in risk assessment, prototyping, and iterative development. Inexperienced teams may struggle to navigate the complexities involved.

Potential for Scope Creep: The iterative nature of the Spiral Model can lead to scope creep if not managed effectively. Each iteration might introduce new features or changes, potentially impacting project timelines and budgets.

Applications of Spiral Model:

Large and Complex Projects: The Spiral Model is particularly suitable for large and complex projects where risks are high. By addressing risks early and iteratively refining the product, teams can mitigate potential challenges and ensure the project's success.

Projects with Changing Requirements: For projects where requirements are likely to evolve or change over time, the Spiral Model offers flexibility through its iterative approach. Stakeholders can provide feedback at each stage, allowing for adjustments based on changing needs.

Mission-Critical Systems: In industries such as aerospace, healthcare, or defense, where reliability, safety, and quality are paramount, the Spiral Model can be beneficial. By systematically addressing risks and incorporating feedback, teams can develop robust and reliable systems.

Innovative Products: For projects focused on developing innovative products or solutions where the requirements are not entirely clear upfront, the Spiral Model provides a framework for exploration, prototyping, and refinement. This iterative approach allows teams to experiment, learn, and adapt based on feedback and results.

Long-Term Projects: The Spiral Model can be effective for long-term projects where requirements, technologies, or market conditions may change over time. By regularly revisiting and reassessing the project's objectives and risks, teams can maintain alignment with stakeholders' expectations and market demands.

SDLC Iterative Model

What is Iterative Model?

The popular iterative model gives an exact performance of the development of software as a life cycle. It primarily focuses on preliminary growth and design and then gains momentum slowly with more complexity as well as meet requirements until the final software is built entirely. So, basically, the iterative development model is an approach of segmenting any large software development process into smaller portions.

This type of SDLC model does not target to establish a complete specification plan. As an alternative, this model is dedicatedly designed to start with minimum requirements specifying as well as implementing only a part of the software. The prototype is then further reviewed for additional requirements. The practice then takes an iterative form to create a new version of the application.

Phases of Iterative Model

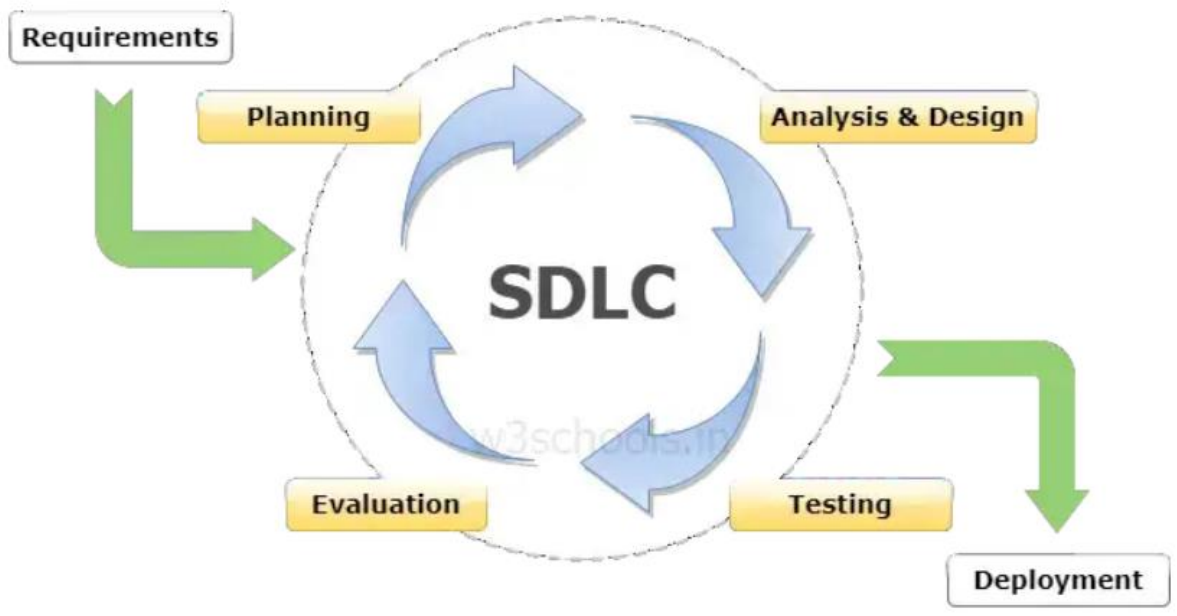


Fig: SDLC Iterative Model

1. **Requirements Phase:** In the requirements phase of software development, the system related information is gathered and analyzed. The collected requirements are then planned accordingly for developing the system.
2. **Design Phase:** In the Design phase, the software solution is prepared to meet the necessities for the design. The system design may be a new one or the extension of a previous build one.
3. **Implementation and Test:** In the implementation as well as a test phase, the system is developed by coding and building the user interface and modules which is then incorporated and tested.
4. **Review Phase:** The review phase is where the software is estimated and checked as per the current requirement. Then, further requirements are reviewed discussed and reviewed to propose for an update in the next iteration.

Advantages of Iterative Model:

Flexibility: The Iterative Model allows for flexibility as changes can be made to the software requirements or design during each iteration based on feedback from stakeholders or testing results.

Early Feedback: Stakeholders can provide feedback early in the development process, allowing for adjustments and refinements to be made before the final product is delivered.

Risk Management: By breaking the project into smaller iterations and addressing high-risk components first, teams can identify and mitigate risks early in the development cycle.

Incremental Delivery: The Iterative Model facilitates incremental delivery of the software, allowing users to start benefiting from the system's functionality sooner rather than waiting for the entire project to be completed.

Continuous Improvement: With each iteration, the software evolves based on feedback and lessons learned from previous iterations, leading to a more refined and robust final product.

Disadvantages of Iterative Model:

Complexity: Managing multiple iterations simultaneously can introduce complexity, requiring careful planning, coordination, and communication among team members and stakeholders.

Potential for Scope Creep: Without proper oversight and control, the scope of the project can expand beyond the initial requirements, leading to increased costs and timelines.

Resource Intensive: The Iterative Model may require more resources, including time and manpower, compared to other SDLC models due to the need for continuous iterations and refinements.

Documentation Overhead: Maintaining documentation for each iteration and managing changes can be challenging, potentially leading to documentation overhead and inconsistencies.

Dependency Issues: Dependencies between iterations or components can introduce challenges, particularly if changes made in one iteration impact others, leading to potential delays or rework.

Applications of Iterative Model:

Projects with Evolving Requirements: The Iterative Model is well-suited for projects where requirements are not fully defined upfront or are likely to change over time. By adopting an iterative approach, teams can adapt to changing needs and priorities more effectively.

Complex Systems Development: For complex systems or projects that involve multiple stakeholders, technologies, or components, the Iterative Model allows for incremental development and refinement, reducing risks and ensuring alignment with stakeholders' expectations.

Innovative and R&D Projects: For projects focused on innovation, research, or development of new products or technologies, the Iterative Model provides a structured yet flexible framework for experimentation, learning, and adaptation based on feedback and results.

Customer-Centric Projects: In projects where user feedback and satisfaction are paramount, such as consumer-facing applications or products, the Iterative Model allows for continuous user involvement and validation, ensuring that the final product meets user needs and expectations.

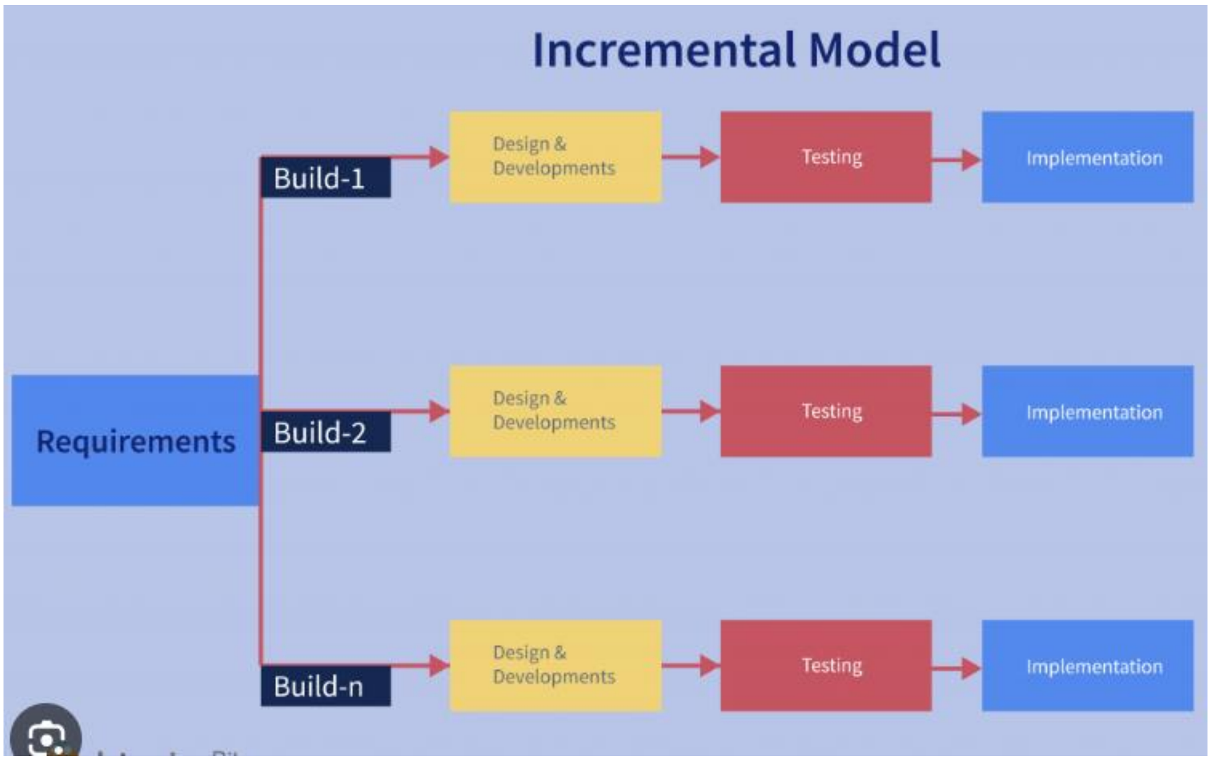
Long-Term Projects: For long-term projects where market conditions, technologies, or requirements may change over time, the Iterative Model provides a responsive and adaptive approach, allowing teams to prioritize and address high-value features or components iteratively based on evolving needs and priorities.

SDLC Incremental Model

What is Incremental Model?

The incremental process model is also known as the Successive version model. First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.

Phases of Incremental Model:



Requirements Gathering: In this initial phase, stakeholders' requirements are gathered and analyzed to define the scope, objectives, and constraints of the project. The requirements are documented and prioritized based on their importance and feasibility.

Design: Once the requirements are defined, the system architecture and design are developed. In the Incremental Model, the design phase may be divided into increments, where each increment focuses on a specific component or module of the system.

Implementation: During this phase, the software is developed based on the design specifications and requirements gathered. The implementation phase is divided into multiple increments, with each increment focusing on developing specific functionalities or features of the system.

Testing: After each increment is developed, it undergoes rigorous testing to identify and fix defects, ensure functionality, and verify compliance with the requirements. Testing is conducted iteratively for each increment, allowing for early detection and resolution of issues.

Deployment: Once an increment is tested and approved, it is deployed or released to users. Users can start using the system or application, providing feedback for further refinements and enhancements.

Feedback and Evaluation: After deployment, users provide feedback on the system's functionality, performance, and usability. Based on this feedback, improvements and enhancements are planned for subsequent increments.

Incremental Development: The development process continues iteratively, with each increment building upon the previous one. New features or functionalities are added, and existing ones are refined based on feedback and requirements.

Maintenance and Support: After all increments are developed and deployed, the system enters the maintenance phase, where ongoing support, updates, and enhancements are provided based on user feedback, changing requirements, or technological advancements.

Advantages:

Early Delivery and Feedback: The Incremental Model allows for early delivery of functional components or modules, enabling users to provide feedback and validate the system's functionality at an early stage.

Reduced Risks: By developing and testing the system in increments, the Incremental Model helps in identifying and mitigating risks early in the development process, leading to more predictable outcomes.

Flexibility: The Incremental Model offers flexibility as changes can be incorporated into subsequent increments based on user feedback, evolving requirements, or technological advancements.

Improved Quality: With each increment undergoing rigorous testing and validation, the Incremental Model helps in ensuring the quality and reliability of the software, leading to a more robust and stable final product.

Cost-Efficient: The Incremental Model can be cost-efficient as it allows for incremental development and deployment, enabling organizations to allocate resources more effectively and prioritize high-value features or components.

Disadvantages of Incremental Model:

Complexity: Managing multiple increments, dependencies, and integration points can introduce complexity, requiring careful planning, coordination, and communication among team members and stakeholders.

Potential for Scope Creep: Without proper oversight and control, the scope of the project can expand beyond the initial requirements, leading to increased costs, timelines, and resource constraints.

Integration Challenges: Integrating different increments or components can be challenging, particularly if changes made in one increment impact others, leading to potential delays, conflicts, or rework.

Documentation Overhead: Maintaining documentation for each increment, managing changes, and ensuring consistency across increments can be challenging, potentially leading to documentation overhead and inconsistencies.

Applications:

Large and Complex Projects: The Incremental Model is well-suited for large and complex projects where requirements are not fully defined upfront or are likely to evolve over time. By adopting an incremental approach, teams can manage complexity, mitigate risks, and ensure alignment with stakeholders' expectations.

Projects with Changing Requirements: For projects where requirements are expected to change or evolve, such as in dynamic business environments or industries, the Incremental Model provides flexibility and adaptability, allowing teams to prioritize and address high-value features or components iteratively.

Long-Term Projects: For long-term projects where market conditions, technologies, or requirements may change over time, the Incremental Model offers a responsive and adaptive approach, enabling organizations to deliver value incrementally and maintain alignment with evolving needs and priorities.

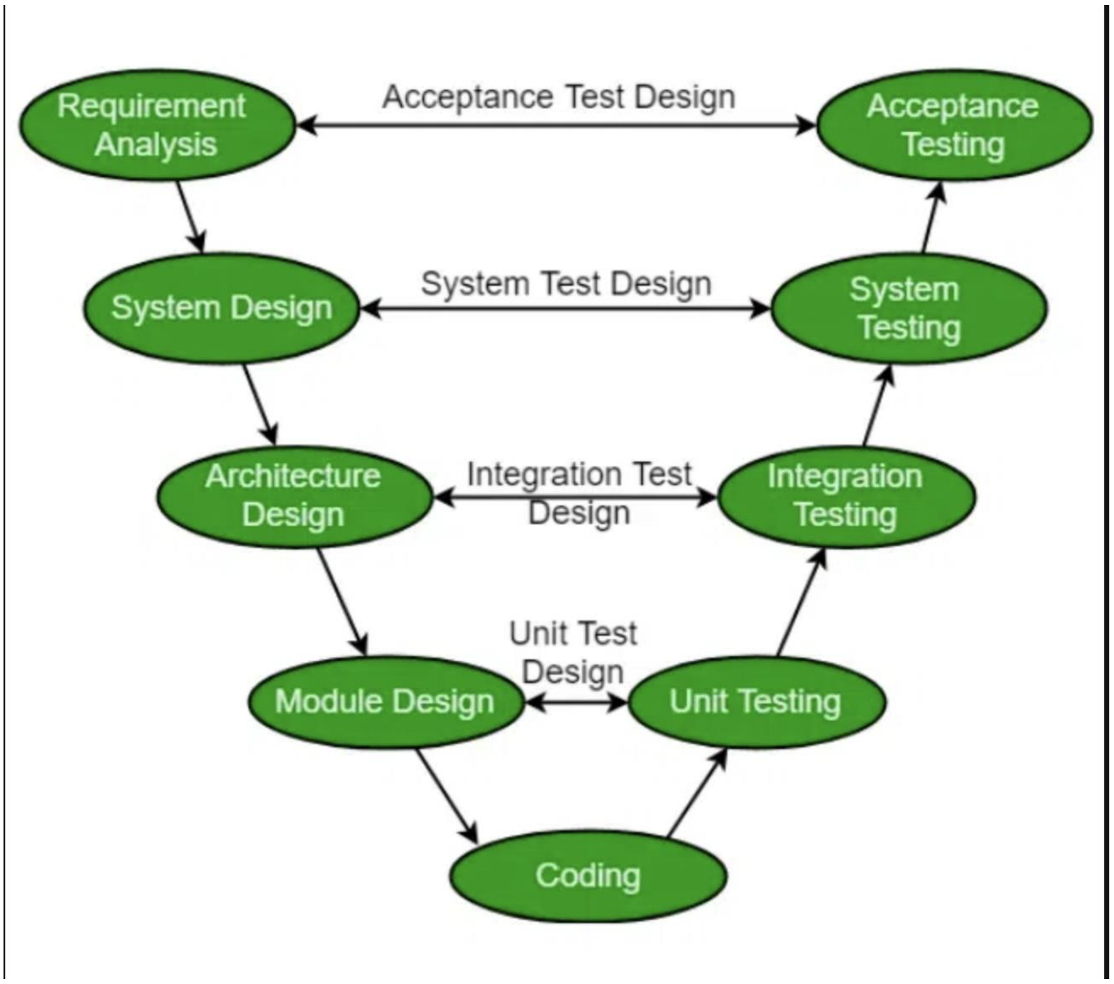
Customer-Centric Projects: In projects where user feedback and satisfaction are paramount, such as consumer-facing applications or products, the Incremental Model allows for continuous user involvement, validation, and refinement, ensuring that the final product meets user needs and expectations.

5) V – model

Features:

- **Requirements Gathering and Analysis:** The first phase of the V-Model is the requirements gathering and analysis phase, where the customer’s requirements for the software are gathered and analyzed to determine the scope of the project.
- **Design:** In the design phase, the software architecture and design are developed, including the high-level design and detailed design.
- **Implementation:** In the implementation phase, the software is actually built based on the design.
- **Testing:** In the testing phase, the software is tested to ensure that it meets the customer’s requirements and is of high quality.

- **Deployment:** In the deployment phase, the software is deployed and put into use.
- **Maintenance:** In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.
- **The V-Model is often used in safety:** critical systems, such as aerospace and defence systems, because of its emphasis on thorough testing and its ability to clearly define the steps involved in the software development process.



Advantages:

1. This is a highly disciplined model and Phases are completed one at a time.
2. V-Model is used for small projects where project requirements are clear.
3. Simple and easy to understand and use.
4. This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
5. It enables project management to track progress accurately.
6. Clear and Structured Process: The V-Model provides a clear and structured process for software development, making it easier to understand and follow.
7. Emphasis on Testing: The V-Model places a strong emphasis on testing, which helps to ensure the quality and reliability of the software.
8. Improved Traceability: The V-Model provides a clear link between the requirements and the final product, making it easier to trace and manage changes to the software.

9. Better Communication: The clear structure of the V-Model helps to improve communication between the customer and the development team.

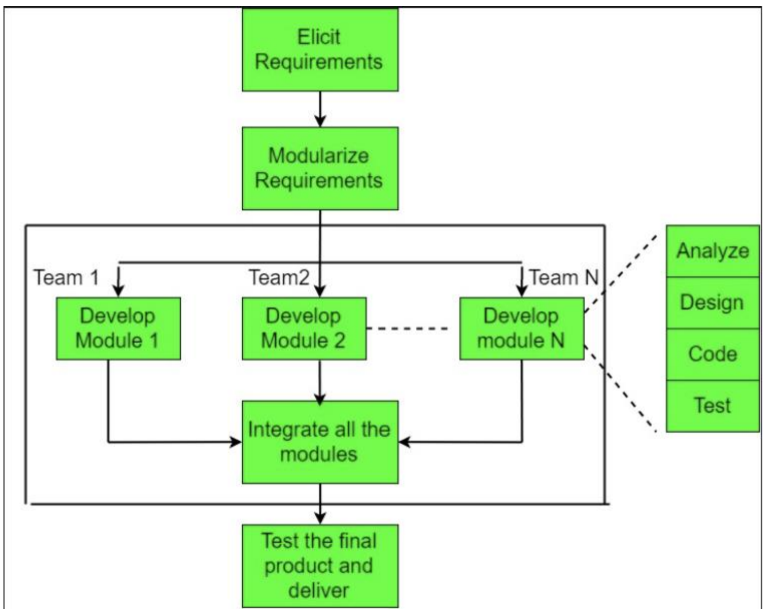
Disadvantages:

1. High risk and uncertainty.
2. It is not a good for complex and object-oriented projects.
3. It is not suitable for projects where requirements are not clear and contains high risk of changing.
4. This model does not support iteration of phases.
5. It does not easily handle concurrent events.
6. Inflexibility: The V-Model is a linear and sequential model, which can make it difficult to adapt to changing requirements or unexpected events.
7. Time-Consuming: The V-Model can be time-consuming, as it requires a lot of documentation and testing.
8. Overreliance on Documentation: The V-Model places a strong emphasis on documentation, which can lead to an overreliance on documentation at the expense of actual development work.

Why preferred?

- It is easy to manage due to the rigidity of the model. Each phase of V-Model has specific deliverables and a review process.
- Proactive defect tracking – that is defects are found at early stage.

2) RAD (Rapid Application Development) model:



Advantages:

1. The use of reusable components helps to reduce the cycle time of the project.
2. Feedback from the customer is available at the initial stages.
3. Reduced costs as fewer developers are required.
4. The use of powerful development tools results in better quality products in comparatively shorter time spans.
5. The progress and development of the project can be measured through the various stages.

Disadvantages:

1. The use of powerful and efficient tools requires highly skilled professionals.
2. The absence of reusable components can lead to the failure of the project.
3. The team leader must work closely with the developers and customers to close the project on time.
4. The systems which cannot be modularized suitably cannot use this model.
5. Customer involvement is required throughout the life cycle.
6. It is not meant for small-scale projects as in such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.

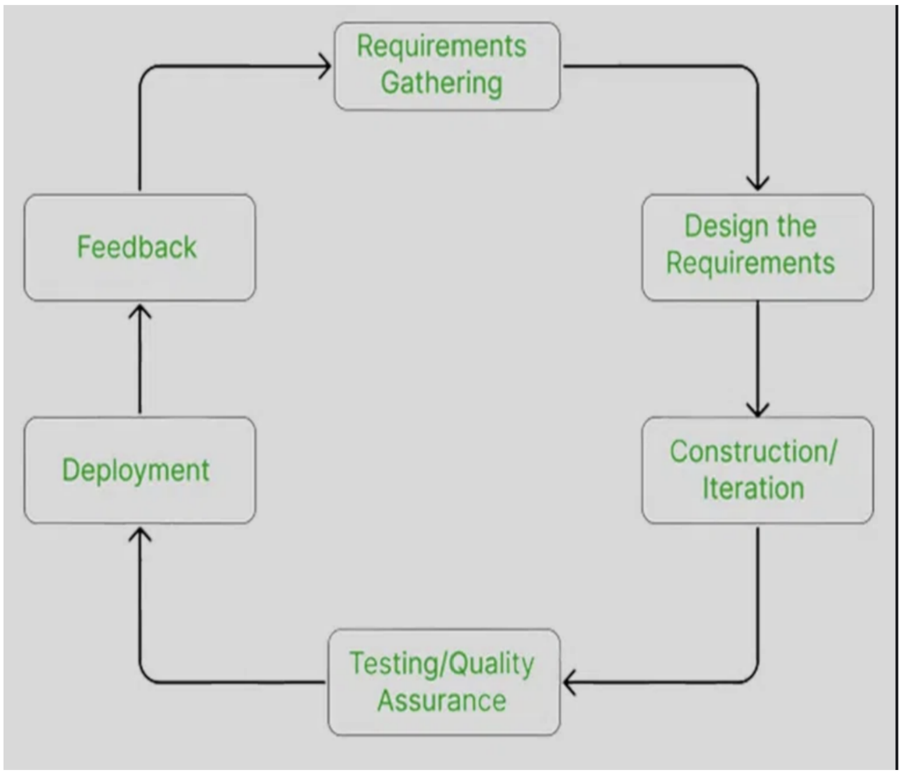
Applications:

1. This model should be used for a system with known requirements and requiring a short development time.
2. It is also suitable for projects where requirements can be modularized and reusable components are also available for development**.**

3) Agile model:

Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)



Advantages:

1. Working through Pair programming produces well-written compact programs which have fewer errors as compared to programmers working alone.
2. It reduces the total development time of the whole project.
3. Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
4. Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
5. Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.

Disadvantages:

1. The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
2. It is not suitable for handling complex dependencies.
3. The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.

Principles of model:

1. The agile model relies on working software deployment rather than comprehensive documentation.
2. Frequent delivery of incremental versions of the software to the customer representative in intervals of a few weeks.
3. Requirement change requests from the customer are encouraged and efficiently incorporated.

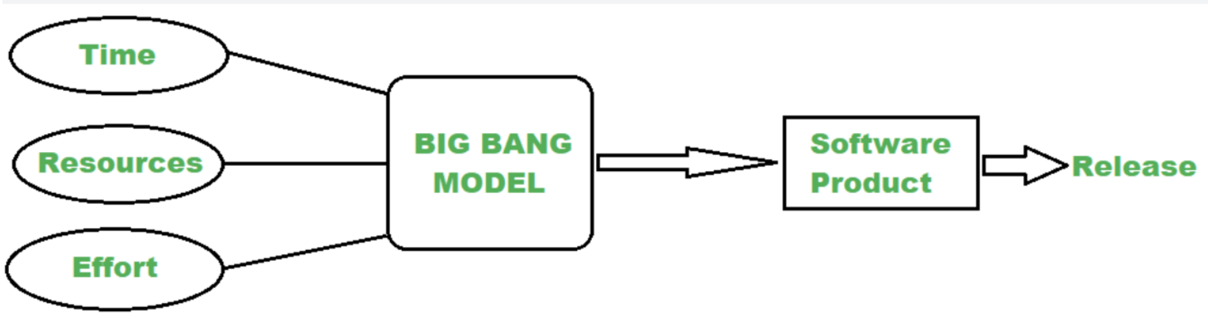
When to use the model ?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

4) Big Bang model:

In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.



When to use the model ?

1. Developing a project for learning purposes or experiment purposes.
2. No clarity on the requirements from the user side.
3. When newer requirements need to be implemented immediately.
4. Changing requirements based on the current developing product outcome.

Features:

1. Not require a well-documented requirement specification
2. Provides a quick overview of the prototype

3. Needs little effort and the idea of implementation
4. Allows merging of newer technologies to see the changes and adaptability

Advantages:

1. There is no planning required for this.
2. Suitable for small projects
3. Very few resources are required.
4. As there is no proper planning hence it does not require managerial staffs
5. Easy to implement

Disadvantages:

1. Not suitable for large projects.
2. Highly risky model and uncertain
3. Might be expensive if requirements are not clear